Proyecto de Fin de Ciclo: Aplicación de Gestión de Videojuegos

David Bargalló Ortiz
Ciclo Formativo de Grado Superior: Desarrollo de Aplicaciones Multiplataforma (DAM)
Curso 2024/2025

Índice

1. Introducción

Este documento recoge el desarrollo del TFG, que es en una aplicación de escritorio para la gestión de colecciones de videojuegos. El objetivo de la aplicación es facilitar a los usuarios la organización, seguimiento y análisis de su biblioteca personal de videojuegos, ya sean físicos o digitales.

Vamos a listar algunas aplicaciones parecidas que existen. Si bien es cierto, que a nivel escritorio no hay mucha oferta, estas aplicaciones/webs pueden ser similares y cubrir necesidades que buscamos cubrir con nuestra app, y podemos tomarlas como ejemplo, tanto como inspiración como para intentar mejorarlas. Por mencionar las más importantes: GG App, HowLongToBeat o Backloggd, las 3 son aplicaciones que permiten llevar un seguimiento de videojuegos. Por ejemplo, una cosa que mejoraremos es el permitir almacenar juegos físicos con su localización. Además nos inspiraremos en estas aplicaciones en el tema de las estadísticas o los precios de los mismos entre otras cosas.

---

2. Descripción del problema

Muchos jugadores acumulan videojuegos en diferentes formatos y plataformas, lo que dificulta su organización. Esta aplicación busca centralizar toda esta información en un único lugar, resolviendo problemas como:

- Desorganización de juegos físicos y digitales.
- Falta de seguimiento de DLCs y complementos.
- Dificultad para localizar juegos físicos.
- Falta de estadísticas de los juegos que ha jugado.
- No poder compartir la colección de juegos completa de todas las plataformas.
- Conocer el precio de sus juegos y tener una wishlist

---

3. Objetivos

Objetivo general

Desarrollar una aplicación de escritorio que permita a los usuarios gestionar su biblioteca de videojuegos de manera completa e intuitiva, además de poder seguir videojuegos que les interese con su precio.

Objetivos específicos

- Crear un sistema de login.
- Permitir añadir, editar, eliminar y filtrar videojuegos (por diferentes parámetros).
- Gestionar wishlist de juegos con seguimiento de precios.
- Mostrar estadísticas de la persona (género más jugado, consola más jugada...).
- Llevar un registro de la uubicación física de cada juego.
- Exportar la colección en formato PDF.
- Conectar con APIs externas para importar datos o precios.

Requisitos funcionales:

- La aplicación debe permitir al usuario buscar videojuegos y añadirlos o eliminarlos de su lista/wishlist
- Consultar los precios de los juegos tanto de la wishlist como los que tiene
- Consultar estadísticas basadas en su biblioteca (de qué género tiene más juego, en qué consola...)

Requisitos no funcionales

- Conexión con API's externas tanto para los precios como para los juegos
- Que el usuario, al buscar un juego por ejemplo, por su nombre, los juegos que se le muestre sean sacados de la API, y si lo añade a su lista, que los datos se extraigan de la api y se introduzcan en la BBDD.
- Log in seguro, contraseña hasheada
- Aplicación escalable.

---

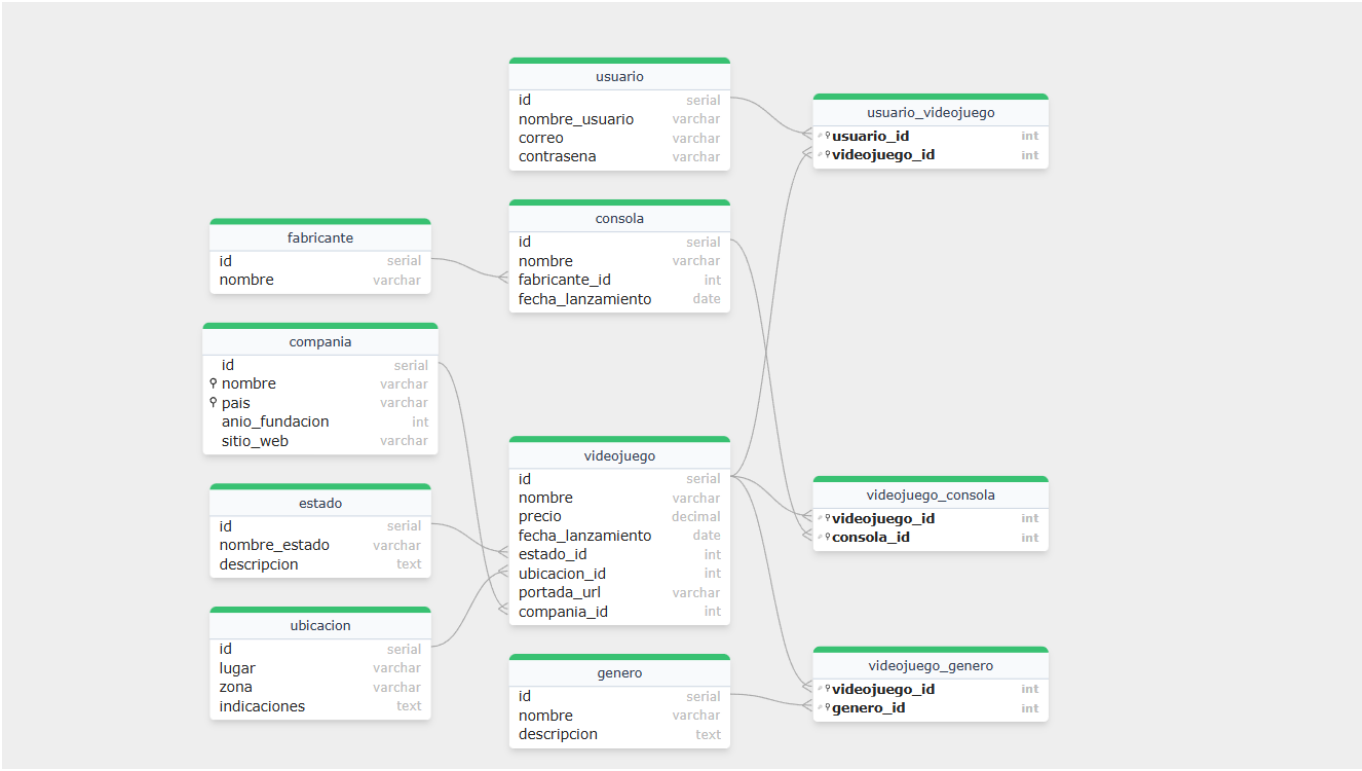
4. Diseño de la solución

4.1 Arquitectura general

La aplicación está dividida en varias capas siguiendo el patrón MVC (Modelo-Vista-Controlador), integrando además Spring Boot que se inicializa al arrancar la aplicación JavaFX. Esto permite separar la lógica, la interfaz de usuario y el acceso a datos.

4.2 Diseño de la base de datos

La base de datos está alojada en Neon (PostgreSQL en la nube). Aquí está el ER del proyecto:



4.3 Diseño de la interfaz

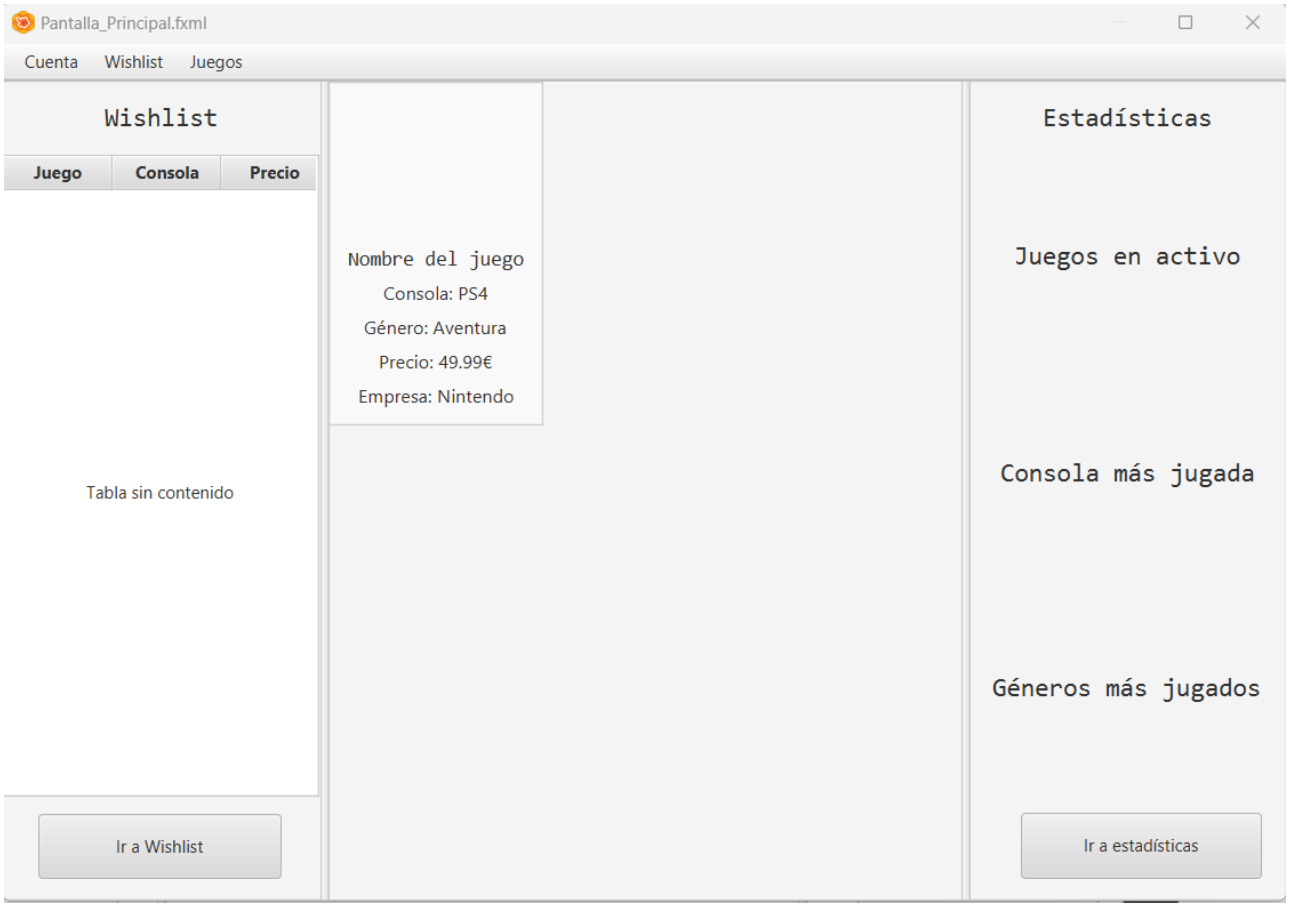
Las interfaces gráficas se han desarrollado con JavaFX y Scene Builder. La navegación entre pantallas se realiza mediante controladores que cargan FXML y reciben datos del backend.

Pantallas incluidas:

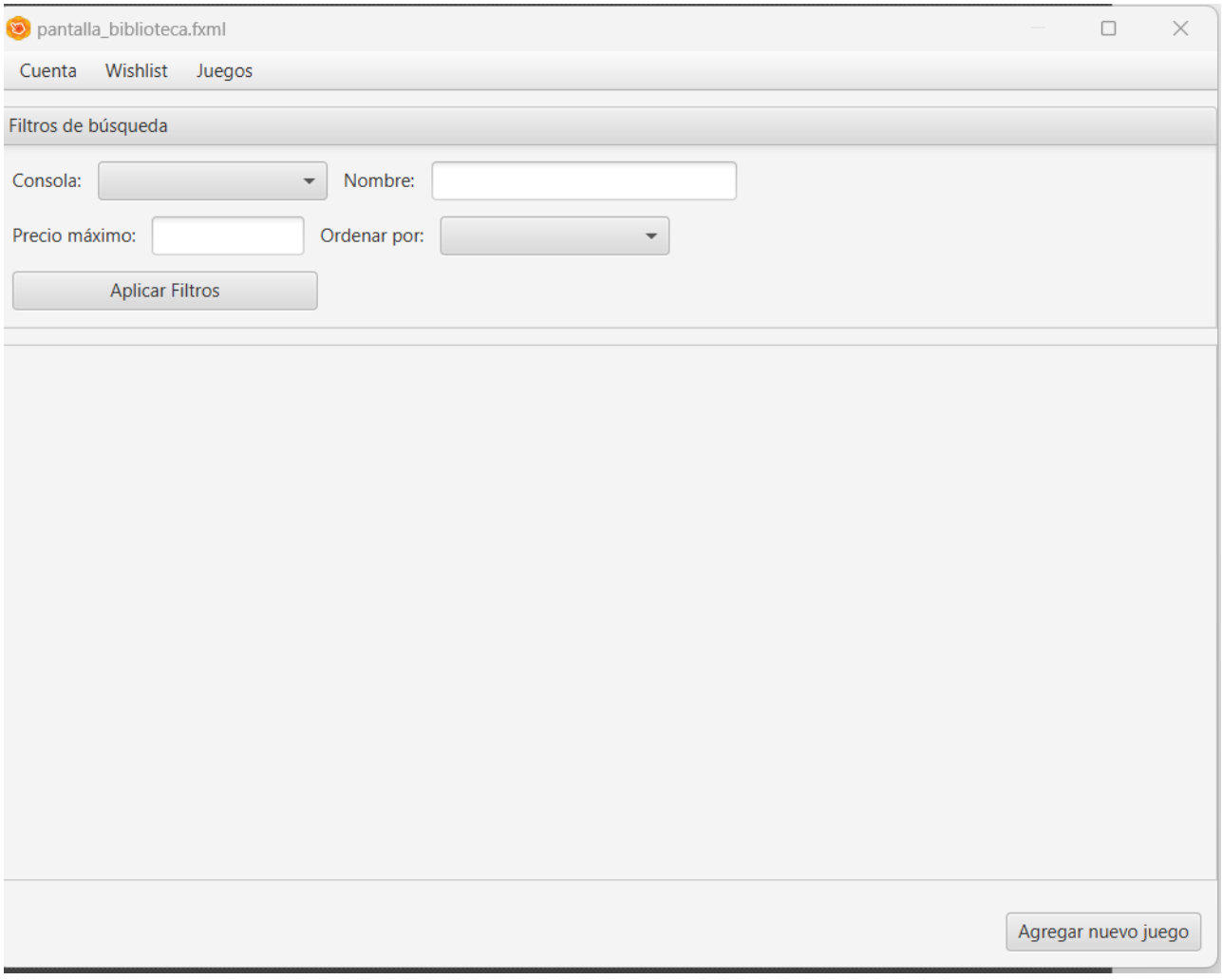
- Inicio de sesión

- Pantalla de registro

- Pantalla principal



- Biblioteca de juegos



- Ficha detallada de cada juego **PENDIENTE DE REALIZAR MODIFICACIONES**
- Wishlist

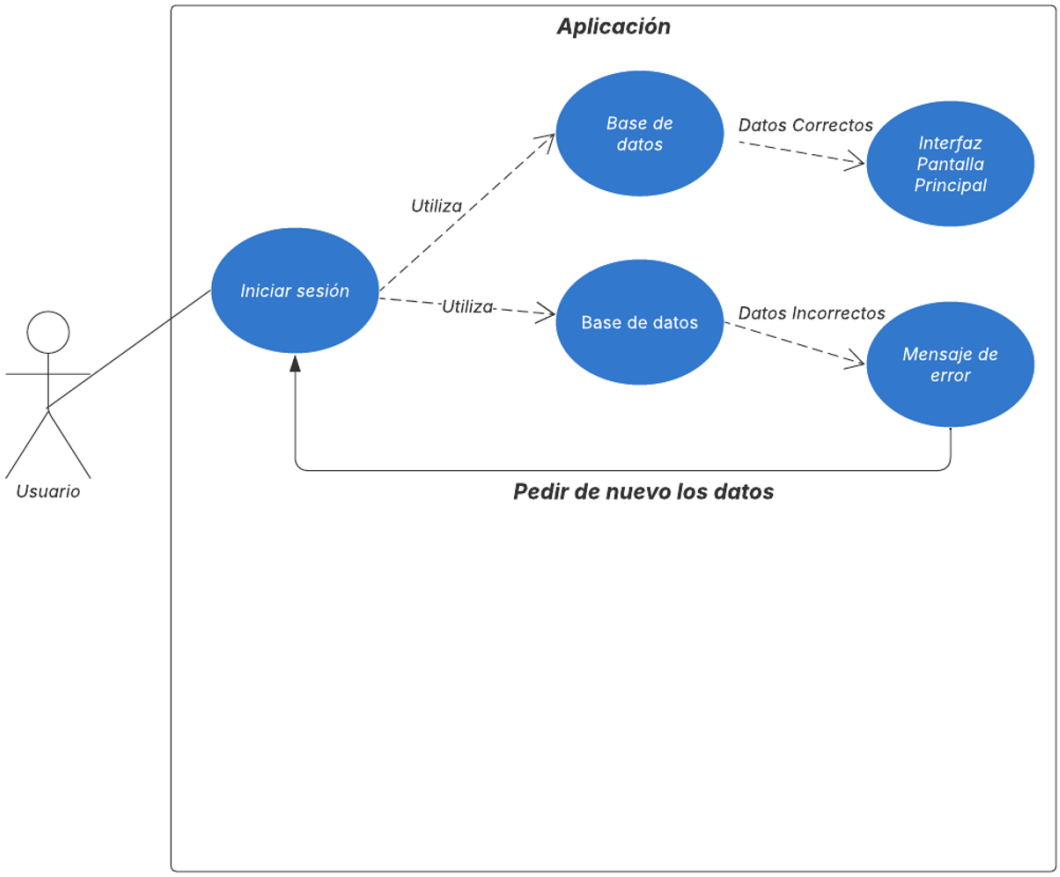
- Estadísticas **PENDIENTE DE REALIZAR MODIFICACIONES**
- Cambio de contraseña **PENDIENTE**

Los filtros estarán dentro de las pantallas de Biblioteca de juegos y Wishlist, para filtrar los juegos. La pantalla principal será una pantalla que muestre algunas estadísticas a modo de preview, igual con los juegos tanto de la biblioteca como de la wishlist

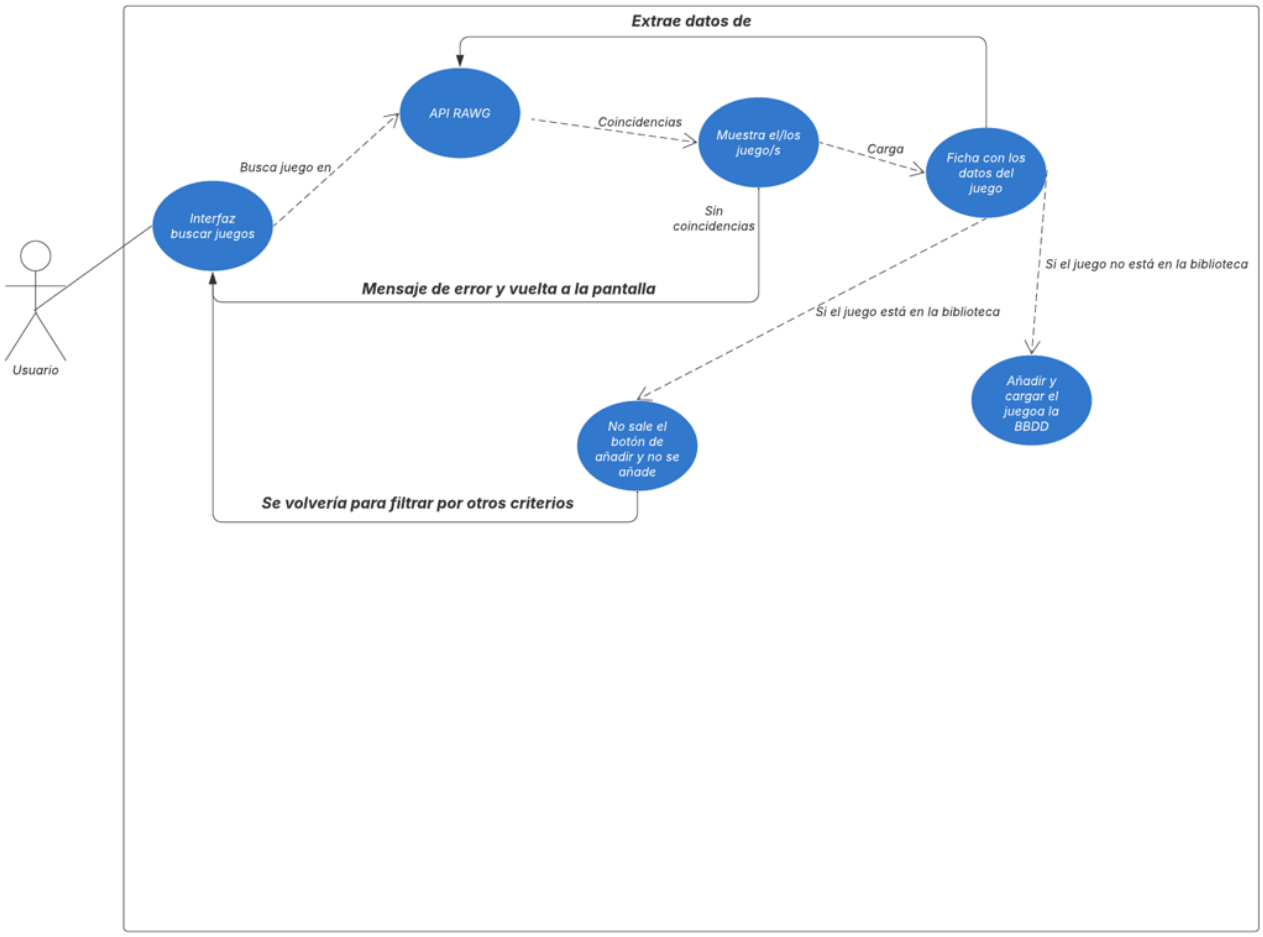
Capturas de las pantallas cuando estén terminadas

4.4 Casos de uso

Caso de uso N1: Inicio de sesión:



Caso de uso N2: Añadir videojuegos a la biblioteca: *



*. Comportamiento similar en el caso de uso de añadir juego a wishlist.

4.5 Justificación de decisiones técnicas

En la fase de análisis del proyecto, se han determinado qué programas o tecnologías se van a usar para el desarrollo del proyecto, esto para poder conseguir los objetivos que se buscan con esta aplicación de la mejor forma posible. Vamos a explicar el razonamiento usado para elegir estas tecnologías, programas etc en lugar de otros:

Java como lenguaje principal

Se eligió Java por ser el lenguaje base del ciclo formativo, también es uno de los lenguajes más usados, por lo tanto, tiene una gran comunidad para consultar posibles errores. Además, su orientación a objetos y el soporte a largo plazo de versiones como Java 17 lo hacen ideal para aplicaciones de escritorio, como el caso de esta aplicación.

JavaFX para la interfaz gráfica

JavaFX ofrece una interfaz más moderna y flexible que Swing, con mejor soporte para CSS, animaciones y componentes personalizados. También permite trabajar de forma visual con Scene Builder, lo que facilita el diseño de interfaces complejas sin necesidad de código extenso. La integración con FXML permite una separación clara entre vista y lógica, siguiendo el patrón MVC.

Spring Boot como backend local

Si bien es cierto que Spring Boot suele ser mayormente usado en aplicaciones web, en este proyecto lo vamos a usar para manejar la lógica de negocio y la conexión a la base de datos, la cual estará en linea, y no a nivel local. Algunos de los puntos fuertes para utillizar Spring Boot en una aplicación de escritorio son los siguientes:

- La facilidad que ofrece para estructurar la aplicación en capas (modelo, servicio, repositorio, controlador).
- Lo sencillo que es para trabajar con JPA para bases de datos relacionales (nuestro caso)
- Una línea de desarrollo a futuro, nos permitiría ampliar la aplicación a una arquitectura cliente-servidor gracias a la modularidad de Spring Boot.

PostgreSQL en la nube con Neon

Aunque se podría haber usado para una aplicación local una base de datos local como SQLite, se ha decidido una base de datos remota, en este caso, Neon, que usa PostgreSQL por varias razones:

- Permite acceder a los datos desde diferentes dispositivos si fuese necesario.
- PostgreSQL es una base de datos potente, robusta y bien soportada, adecuada para proyectos que podrían crecer en volumen de datos o funcionalidad.
- En caso de, por ejemplo, tener que reinstalar la aplicación, o el SO, no se perdería la base de datos con toda su información.

APIs externas (RAWG y PriceCharting)

Estas APIs permiten obtener datos actualizados y precisos sobre videojuegos y precios, sin necesidad de construir una base de datos propia desde cero. Su uso se justifica por:

Con estas API's, conseguimos no tener que hacer al usuario añadir manualmente todos los datos (solamente algunos como la ubicación física), además de datos actualizados como los precios. Algunos de los motivos más específicos son:

- Datos en tiempo real.
- Permite funcionalidades como la wishlist con precios.
- Permite centrarse en la lógica de negocio sin necesidad de mantener un repositorio interno de juegos.

Maven como gestor de dependencias

Maven facilita la gestión de librerías y versiones, automatizando el proceso de construcción y organización del proyecto. Es especialmente útil cuando se trabaja con frameworks como Spring Boot o JavaFX, que tienen múltiples dependencias.

GitHub para control de versiones

Se utiliza GitHub para llevar un control del desarrollo, permitir volver a una versión anterior en caso de error, y tener todos los cambios en el proyecto documentados. Además de tener el proyecto subido en línea, por si en el futuro se convierte en uno colaborativo.

---

5. Desarrollo técnico

5.1 Tecnologías utilizadas

- **Lenguaje principal:** Java
- **Frameworks:** JavaFX, Spring Boot
- **ORM / BBDD:** Neon (PostgreSQL vía REST)
- **Diseño UI:** JavaFX + Scene Builder
- **Gestor de dependencias:** Maven
- **Control de versiones:** Git + GitHub
- **APIs externas:** RAWG, PriceCharting
- **Pruebas:** JUnit, Postman

5.2 Estructura del proyecto

El proyecto está organizado en paquetes:

- `modelo`: entidades del dominio
- `servicio`: lógica de negocio
- `repositorio`: acceso a base de datos
- `controlador`: controladores de cada pantalla
- `vista`: archivos FXML
- `utilidades`: clases auxiliares para hashing, exportación PDF, validación, etc.

5.3 Funcionalidades implementadas

- Inicio de sesión y gestión de usuarios.
- Conexión a la BBDD
- CRUD de videojuegos con datos detallados. **FALTAN PRUEBAS**
- Pantallas visuales intuitivas mediante JavaFX.

Hay algunas funcionalidades avanzadas, pero aún no están terminadas al 100%, por lo tanto no son incluídas aquí.

---

6. Pruebas

LAS PRUEBAS AÚN NO SE HAN LLEVADO A CABO, PUES SERÁN AL FINAL.

- Pruebas unitarias con JUnit.
- Pruebas funcionales: navegación, filtros, exportación.
- Pruebas de carga con colecciones extensas.
- Pruebas de conexión con la Base de datos.
- Pruebas de conexión con APIs externas.
- Pruebas manuales por parte de usuarios para feedback.

---

7. Manual de usuario

Requisitos

- Sistema operativo: Windows
- Java 22
- Conexión a Internet (opcional para usar APIs)

Pasos básicos

1. Iniciar sesión o registrarse.
2. Añadir juegos a la biblioteca.
3. Buscar juegos con filtros.
4. Ver wishlist y actualizar precios.
5. Exportar la colección a PDF.

MANUAL DE USUARIO EN DESARROLLO, FALTAN CAPTURAS Y TUTORIAL

---

8. Manual técnico

Requisitos de desarrollo

- Java 22
- Maven
- Visual Studio Code
- Cuenta en Neon

- Claves API para RAWG y PriceCharting

Pasos para ejecutar

1. Clonar repositorio: `git clone https://github.com/DavidBargallo/TFG`
2. Ejecutar `App.java` (Usando Maven mvn javafx:run).
3. Verificar que Spring Boot arranca correctamente.

---

9. Conclusiones

SE AÑADIRÁ CUANDO SE TERMINE EL PROYECTO

---

10. Bibliografía

- OpenAI. (2025). *ChatGPT*. Para consultas de errores y código básico Recuperado de https://chatgpt.com/
- Google. (2025). *Google Search*. Para buscar diferentes informaciones Recuperado de https://www.google.com/
- Gemini. (2025). *Gemini AI*. Para consultas de errores, usabilidad de interfaces y código básico Recuperado de https://gemini.google.com/
- Stack Overflow. (2025). *Stack Overflow*. Para preguntar y buscar respuestas en los foros Recuperado de https://stackoverflow.com/
- Neon. (n.d.). *Neon Docs*. Para consultas de cómo conectar la app a su BBDD y realizar operaciones Recuperado de https://neon.tech/docs
- Oracle. (n.d.). *Java Documentation*. Para dudas básicas de programación en java Recuperado de https://docs.oracle.com/en/java/
- RAWG. (n.d.). *RAWG Video Games Database API*. Para consultar lo que ofrece la aplicación en cuanto a datos y cómo usarla Recuperado de https://rawg.io/apidocs
- Scene Builder. ()

---

11. Anexos

PENDIENTE